View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by CWI's Institutional Repository

Science of Computer Programming 78 (2013) 2192–2215

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Stream processing coalgebraically

Milad Niqui [a], Jan J.M.M. Rutten [a,b,*]

[a] Centrum Wiskunde & Informatica (CWI), The Netherlands

[b] Radboud University Nijmegen, The Netherlands

ARTICLE INFO

Article history:
Received 7 January 2011
Received in revised form 8 September 2011
Accepted 28 January 2012
Available online 27 July 2012

MSC:
11B85
68Q70
68Q85
11B57

Keywords:
Stream calculus
Dataflow programming
Coinduction
Rational stream
Algebraic stream
Stream circuit
Moessner's theorem

ABSTRACT

We study various operations for splitting, partitioning, projecting and merging streams of data. These operations are motivated by their use in dataflow programming and stream processing languages. We use the framework of *stream calculus* and *stream circuits* for defining and proving properties of such operations using behavioural differential equations and coinduction proof principles. As a featured example we give proofs of results, observed by Moessner, from elementary number theory using our framework. We study the invariance of certain well patterned classes of streams, namely rational and algebraic streams, under splitting and merging. Finally we show that stream circuits extended with gates for dyadic split and merge are expressive enough to realise some non-rational algebraic streams, thereby going beyond ordinary stream circuits.

© 2013 Published by Elsevier B.V.

1. Introduction

In this paper, we study various operations for splitting, partitioning, projecting and merging streams (infinite sequences of data). These operations are motivated by their use in dataflow programming and stream processing languages (e.g., [4]).

Our perspective on streams and stream operations is essentially coalgebraic. More specifically, we use the framework of *stream calculus* [17] and *stream circuits* [18] for defining and proving properties of such operations. Definitions are typically given using behavioural stream differential equations. Proofs will mostly be given by coinduction, with which two streams can be shown to be equal by the construction of a suitable stream bisimulation relation between them.

The use of stream calculus and coinduction leads to new and simpler definitions and proofs of several existing notions and properties, some of which are taken from [13]. To mention already one example here (see Sections 3 and 5 for more): a periodic stream sampler S is a stream operation that produces a substream of a given stream σ by taking out of each block of $l \geq 0$ elements a subset of $k \leq l$ elements (at fixed positions). Periodic stream samplers can be defined by the following equation

$$S(\sigma)^{(k)} = S(\sigma^{(l)})$$

* Corresponding author.
E-mail addresses: M.Niqui@cwi.nl (M. Niqui), Jan.Rutten@cwi.nl (J.J.M.M. Rutten).

0167-6423/$ – see front matter © 2013 Published by Elsevier B.V.
doi:10.1016/j.scico.2012.07.013

where $(-)^{(i)}$ denotes the i-th stream derivative, which is defined as the operation *tail* applied i times. In addition, one has to specify the first k initial values of $S(\sigma)$, which are projections from the first l elements of the argument stream σ.

The above equation is an instance of what in stream calculus is called a *stream differential equation*. Such equations define streams and operations on streams by specifying their stream derivatives, similarly to the way differential equations define functions in classical calculus by specifying their derivatives.

The differential equation above is elementary, almost trivial. Yet it allows for proofs of basic facts (such as: composing two periodic stream samplers yields again a periodic stream sampler) that, as we shall see, are much simpler than those in the literature.

Using stream calculus and stream circuits, we obtain also a number of new results. More specifically, we prove (in Sections 6 and 7) the invariance of certain well patterned classes of streams, namely rational and algebraic streams, under the operations of splitting and merging. Furthermore, we show (in Section 8) that stream circuits extended with gates for dyadic split and merge are expressive enough to realise some non-rational algebraic streams (such as the Prouhet–Thue–Morse stream), thereby going beyond ordinary stream circuits.

As mentioned above, this paper attempts to give a new perspective on existing notions and results, and also obtains some modest new results. The presented new outlook gives rise to a host of further questions and research directions. Section 9 discusses related work and future research.

2. Preliminaries

In this section, we present the basic definitions on streams that we shall be using. Furthermore, we give a brief overview of a coinductive calculus of streams. We refer the reader to [16,17] for a complete and detailed treatment.

We define the set of *streams* over a set A by $A^\omega = \{\sigma \mid \sigma : \mathbb{N} \to A\}$. We usually use the Greek letters $\sigma, \tau, \ldots$ to denote streams. We denote elements $\sigma \in A^\omega$ by $\sigma = (\sigma(0), \sigma(1), \sigma(2), \ldots)$. The *stream derivative* of a stream σ is $\sigma' = (\sigma(1), \sigma(2), \sigma(3), \ldots)$ and the *initial value* of σ is $\sigma(0)$. For $n \geq 0$ and $\sigma \in A^\omega$, we define higher-order derivatives by $\sigma^{(0)} = \sigma$ and $\sigma^{(n+1)} = (\sigma^{(n)})'$. We have $\sigma(n) = \sigma^{(n)}(0)$.

A *stream bisimulation relation* is a set $R \subseteq A^\omega \times A^\omega$ such that, for all $(\sigma, \tau) \in R$,

$$\sigma(0) = \tau(0) \quad \text{and} \quad (\sigma', \tau') \in R.$$

We write $\sigma \sim \tau$ if there exists a bisimulation R with $(\sigma, \tau) \in R$ and say that σ and τ are bisimilar. The *coinduction proof principle* allows us to prove the equality of two streams by establishing the existence of an appropriate bisimulation relation.

Theorem 1 (*Coinduction*). *For all $\sigma, \tau \in A^\omega$,*

$$\sigma \sim \tau \;\Rightarrow\; \sigma = \tau.$$

Proof. One proves by induction on n that $\sigma(n) = \tau(n)$, for any pair of bisimilar streams σ and τ. □

If A has some algebraic structure, A^ω inherits (parts of) this structure. Assume $\langle A, +, \cdot, -, 0, 1\rangle$ is a ring.[1] For $r \in A$, we define the streams $[r] = (r, 0, 0, 0, \ldots)$ and $\overline{r} = (r, r, r, \ldots)$. We often denote the former simply as r. Another important stream is $X = (0, 1, 0, 0, 0, \ldots)$, whose role will become clear shortly. For $\sigma, \tau \in A^\omega$ and $n \geq 0$, the operations of *sum*, *convolution product* and *Hadamard product* are given by

$$(\sigma + \tau)(n) = \sigma(n) + \tau(n),$$
$$(\sigma \times \tau)(n) = \sum_{i=0}^{n} \sigma(i) \cdot \tau(n-i),$$
$$(\sigma \odot \tau)(n) = \sigma(n) \cdot \tau(n).$$

Since the convolution product corresponds to the product of power series, we simply refer to it as the product. It can be shown that $\langle A^\omega, +, \times, [0], [1]\rangle$ is a ring [3].

Multiplying a stream σ with the constant stream X yields

$$X \times \sigma = (0, \sigma(0), \sigma(1), \sigma(2), \ldots). \tag{2.1}$$

As a consequence, we have that

$$X \times X = (0, 0, 1, 0, 0, 0, \ldots) \quad X \times X \times X = (0, 0, 0, 1, 0, 0, 0, \ldots)$$

and so on. We call a stream $\pi \in A^\omega$ *polynomial* if there are $k \geq 0$ and $a_i \in A$ such that

$$\pi = a_0 + a_1 X + a_2 X^2 + \cdots + a_k X^k = (a_0,\ a_1,\ a_2,\ \ldots,\ a_k,\ 0,\ 0,\ 0,\ \ldots)$$

where we write $a_i X^i$ for $[a_i] \times X^i$ and where X^i denotes the i-fold product of X with itself.

[1] In fact many of the operations on A^ω only need a semiring structure on A [3,20].

One can compute a stream from its initial value and derivative by the so-called *fundamental theorem* of stream calculus [17]: for all $\sigma \in A^\omega$,

$$\sigma = \sigma(0) + (X \times \sigma') \tag{2.2}$$

(writing $\sigma(0)$ for $[\sigma(0)]$), which is easily proved using identity (2.1) above, applied to σ'.

Next assume that A is a field, i.e., every nonzero element has a unique multiplicative inverse. Then this multiplicative inverse operation may be carried over to A^ω [17]: if $\sigma(0) \neq 0$ then the stream σ has a (unique) multiplicative inverse σ^{-1} in A^ω, satisfying $\sigma^{-1} \times \sigma = [1]$. As usual, we shall often write $1/\sigma$ for σ^{-1} and σ/τ for $\sigma \times \tau^{-1}$. Note that the initial value of the sum, product and inverse of streams is given by the sum, product and inverse of their initial values.

If A is a field, a stream $\rho \in A^\omega$ is *rational* if it is the quotient $\rho = \sigma/\tau$ of two polynomial streams σ and τ with $\tau(0) \neq 0$.

The fundamental theorem of stream calculus allows us to solve *stream differential equations* such as $\sigma' = 2 \times \sigma$ with initial value $\sigma(0) = 1$ by computing

$$\begin{aligned} \sigma &= \sigma(0) + (X \times \sigma') \\ &= 1 + (X \times 2 \times \sigma) \end{aligned}$$

which leads to the solution $\sigma = 1/(1 - 2X)$.

A direct consequence of identity (2.1) above is that

$$(X \times \sigma)' = \sigma \tag{2.3}$$

Combining this with the fundamental (2.2), leads to an easy calculation rule for the computation of derivatives:

$$\sigma' = (\sigma - \sigma(0))' \tag{2.4}$$

This identity makes the computation of stream derivatives often surprisingly simple. For instance, for $\sigma = 1/(1 - X)^2$, we have

$$\begin{aligned} \sigma' &= (\sigma - \sigma(0))' \\ &= \left(\frac{1}{(1-X)^2} - 1\right)' \\ &= \left(\frac{2X - X^2}{(1-X)^2}\right)' \\ &= \left(X \times \frac{2 - X}{(1-X)^2}\right)' \\ &= \frac{2 - X}{(1-X)^2}. \end{aligned}$$

For more stream calculations we refer the reader to [17].

In the remainder of the article we assume that A is a field. Strictly speaking, this is not always necessary as some of the constructs, e.g. the stream samplers, do not presume any algebraic structure on A. Nevertheless, in order to be able to freely use the stream calculus we make this assumption. In Section 7 we work in the special case where $A := \mathbb{F}_q$ is a finite field.

3. Periodic stream samplers

Traditionally, a substream of an infinite stream $\sigma : \mathbb{N} \to A$ is defined by means of a (strictly) monotone function $f : \mathbb{N} \to \mathbb{N}$: if $n < m$ then $f(n) < f(m)$. Such an *index function* determines an (infinite) substream $S_f(\sigma)$ by

$$S_f(\sigma)(n) = \sigma(f(n))$$

and conversely, any substream of σ determines a unique such monotone function. Assigning to any stream the substream determined by a given monotone function f defines a *stream sampler*

$$S_f : A^\omega \to A^\omega, \qquad \sigma \mapsto S_f(\sigma).$$

Periodic stream samplers are such that they produce a substream of a given input stream by repeatedly choosing (at fixed intervals) elements from the input stream and ignoring all others. For instance, the function $even : A^\omega \to A^\omega$ given by

$$even(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \ldots)$$

takes of each incoming two elements the first and ignores the second. We say that *even* has *(input) period* 2 and *(output) block size* 1. Another example is the drop operator $D_4^2 : A^\omega \to A^\omega$ given by

$$D_4^2(\sigma) = (\sigma(0), \sigma(1), \sigma(3), \sigma(4), \sigma(5), \sigma(7), \ldots)$$

which drops from each four incoming elements the third and keeps all the others. Note that we always start counting at zero hence $\sigma(2)$, $\sigma(6)$ etc. are dropped. The operator D_4^2 has period 4 and block size 3.

As it turns out, it is somewhat cumbersome to define these and similar such periodic stream samplers by means of monotone index functions. Moreover, it is not entirely trivial to prove simple general facts such as: the composition of two periodic stream samplers is again a periodic stream sampler. Therefore, we prefer the following coinductive definition which uses a stream differential equation.

Definition 2. Let $k, l \in \mathbb{N}$ with $l > 1$ and $1 \leq k \leq l$. Any sequence of k numbers $0 \leq n_0 < n_1 < \cdots < n_{k-1} < l$ determines a *periodic stream sampler* $S : A^\omega \to A^\omega$ of (input) period l and (output) block size k defined by the following stream differential equation:

$$S(\sigma)^{(k)} = S(\sigma^{(l)})$$

with initial values

$$S(\sigma)(j) = \sigma(n_j) \quad (0 \leq j < k).$$

We do not require period and block size to be minimal. If a stream sampler has period l and block size k then it also has period $2l$ with block size $2k$, etc.

The functions *even* and D_4^2 above are given by

$$\begin{aligned} &even(\sigma)' = even(\sigma''), && even(\sigma)(0) = \sigma(0);\\ &D_4^2(\sigma)^{(3)} = D_4^2(\sigma^{(4)}), && D_4^2(\sigma)(0) = \sigma(0),\\ &D_4^2(\sigma)(1) = \sigma(1), && D_4^2(\sigma)(2) = \sigma(3). \end{aligned}$$

Proposition 3. *If $S, T : A^\omega \to A^\omega$ are two periodic stream samplers then so is $T \circ S$.*

Proof. Let S and T satisfy

$$\begin{aligned} &S(\sigma)^{(k)} = S(\sigma^{(l)}), \qquad S(\sigma)(j) = \sigma(n_j) \quad (0 \leq j < k),\\ &T(\sigma)^{(p)} = T(\sigma^{(q)}), \qquad T(\sigma)(j) = \sigma(m_j) \quad (0 \leq j < p). \end{aligned}$$

We claim that $T \circ S$ is a periodic stream sampler with period $l \times q$ and block size $k \times p$. We define a sequence $i_0, i_1, \ldots, i_{q\times k-1}$ by

$$i_{(x\times k)+y} = (x \times l) + n_y \quad (\text{all } x, y \text{ with } 0 \leq x < q, 0 \leq y < k).$$

Next we define a sequence $0 \leq h_0 < h_1 < \cdots < h_{(k\times p)-1} < q \times k$ by

$$h_{(x\times p)+y} = (x \times q) + m_y \quad (\text{all } x, y \text{ with } 0 \leq x < k, 0 \leq y < p).$$

One readily shows that $T \circ S$ satisfies

$$T \circ S(\sigma)^{(k\times p)} = T \circ S(\sigma^{(l\times q)}), \qquad T \circ S(\sigma)(j) = \sigma(i_{h_j}) \quad (0 \leq j < (k \times p) - 1). \quad \square$$

Next we provide some examples by introducing the family of all drop operators.

Definition 4. For $l \geq 2$ and $0 \leq i < l$ we define the drop operator

$$D_l^i : A^\omega \to A^\omega$$

which drops from each input block of size l the i-th element, by the following system of stream differential equations:

$$\begin{aligned} &D_l^{i+1}(\sigma)' = D_l^i(\sigma'), \qquad D_l^{i+1}(\sigma)(0) = \sigma(0) \quad (\text{all } l \geq 2, 0 \leq i < l-1),\\ &D_l^0(\sigma)' = D_l^{l-2}(\sigma''), \qquad D_l^0(\sigma)(0) = \sigma(1) \quad (\text{all } l \geq 2). \end{aligned}$$

Note that for D_4^2, this definition is equivalent with our earlier definition above; also note that $even = D_2^1$.

One of the benefits of coinductive definitions is that they support coinductive proofs. While in the next section we give a featured example of this proof technique, here we prove the so-called *Drop exchange rule* from [13]: for all $l \geq 1$, $0 \leq k \leq h \leq l$,

$$D_{l+1}^h \circ D_{l+2}^k = D_{l+1}^k \circ D_{l+2}^{h+1}.$$

In order to prove this equality, we define a relation $R \subseteq A^\omega \times A^\omega$ by

$$R = \{\langle D_{l+1}^h \circ D_{l+2}^k(\sigma), D_{l+1}^k \circ D_{l+2}^{h+1}(\sigma)\rangle \mid \sigma \in A^\omega\}.$$

The equality now follows by coinduction from the fact that $R \cup R^{-1}$ is a stream bisimulation.

Here is another example. It is a basic instance of a *Drop expansion rule* in [13]:

$$D_2^0 = D_4^0 \circ D_5^2 \circ D_6^4.$$

For a proof, we define a relation $R \subseteq A^\omega \times A^\omega$ by

$$\begin{aligned} R \;&=\; \{\langle D_2^0(\sigma), D_4^0 \circ D_5^2 \circ D_6^4(\sigma)\rangle \mid \sigma \in A^\omega\} \\ &\cup\; \{\langle D_2^0(\sigma), D_4^2 \circ D_5^0 \circ D_6^2(\sigma)\rangle \mid \sigma \in A^\omega\} \\ &\cup\; \{\langle D_2^0(\sigma), D_4^1 \circ D_5^3 \circ D_6^0(\sigma)\rangle \mid \sigma \in A^\omega\}. \end{aligned}$$

The equality follows by coinduction from the fact that R is a stream bisimulation.

Returning to the general question of how to define substreams out of a given stream, we present yet another alternative to the use of monotone index functions, which is also well suited for a coinductive approach. Let $2 = \{0, 1\}$ and let 2^ω be the set of bitstreams. Note that there is a trivial field structure on 2 and hence we can apply stream calculus to 2^ω. Consider a bitstream $\alpha \in 2^\omega$ that is not eventually constant 0, i.e., there is no n such that $\alpha^{(n)} = [0]$. Then for any stream $\sigma \in A^\omega$, α defines a substream $S_\alpha(\sigma)$ consisting of those elements $\sigma(n)$ for which $\alpha(n) = 1$. (Note that the condition on α ensures that $S_\alpha(\sigma)$ is again an infinite stream.) Such a stream α acts as an oracle that tells us of any element of σ whether or not it should be included in the substream we are defining.

More formally, we first note that a stream $\alpha \in 2^\omega$ is eventually constant 0 if it is a polynomial. If α is non-polynomial, it is of the form

$$\alpha = X^n \times (1 + X \times \beta)$$

for some $n \geq 0$ and some $\beta \in 2^\omega$ that is again non-polynomial. Now we define $S_\alpha(\sigma)$ by the following system of differential equations, for arbitrary $\sigma \in A^\omega$ and non-polynomials $\alpha \in 2^\omega$:

$$S_\alpha(\sigma)' = S_\beta(\sigma^{(n+1)}), \qquad S_\alpha(\sigma)(0) = \sigma(n) \quad (\alpha = X^n \times (1 + X \times \beta)).$$

In this manner, any non-polynomial bitstream determines a substream and, conversely, any substream determines a non-polynomial bitstream.

It is now extremely simple to characterise *periodic* stream samplers:

S_α is periodic with period l *if and only if* $\alpha^{(l)} = \alpha$.

The (output) block size is determined by the number of 1's in the set $\{\alpha(0), \ldots, \alpha(l-1)\}$.

Composition of stream samplers can be described in terms of composition of the corresponding oracle bitstreams, which we define as follows.

Definition 5. For all $\alpha, \beta \in 2^\omega$, we define $\beta * \alpha \in 2^\omega$ by the following system of differential equations:

$$(\beta * \alpha)' = \begin{cases} \beta' * \alpha' & \text{if } \alpha(0) = 1 \\ \beta * \alpha' & \text{if } \alpha(0) = 0 \end{cases} \qquad (\beta * \alpha)(0) = \beta(0) \cdot \alpha(0).$$

This composition operator is associative but not commutative and has $1/(1-X)$ as a neutral element: $\sigma * 1/(1-X) = 1/(1-X) * \sigma = \sigma$. It is not difficult to show that

$$S_\beta \circ S_\alpha = S_{\beta * \alpha}.$$

An alternative proof of Proposition 3 is now extremely easy: it follows from the fact that $\alpha^{(n)} = \alpha$ and $\beta^{(m)} = \beta$ imply $(\beta * \alpha)^{(n \times m)} = \beta * \alpha$.

Above we showed some examples of coinductive proof techniques. This can be seen as the advantage of the *coinductive* aspect of our stream calculus. However our stream calculus, being a *calculus*, also allows us to prove identities by calculations. Let us conclude this section with an example illustrating how one can reason about stream sampler composition in terms of stream calculus applied to the corresponding oracle streams. Periodic oracle bitstreams are always of the form

$$\frac{a_0 + a_1 X + a_2 X^2 + \cdots + a_{l-1} X^{l-1}}{1 - X^l}$$

for $a_0, a_1, a_2, \ldots, a_{l-1} \in 2$, not all 0. For our drop operators, for instance, one has

$$D_l^i = S_{\alpha_l^i} \quad \text{with } \alpha_l^i = (1 + X + \cdots + X^{i-1} + X^{i+1} + \cdots + X^{l-1})/(1 - X^l).$$

The equality $D_2^0 = D_4^0 \circ D_5^2 \circ D_6^4$, which we proved above by coinduction, can also be deduced from the following computation in stream calculus on the corresponding oracle bitstreams:

$$\begin{aligned} \alpha_4^0 * \alpha_5^2 * \alpha_6^4 &= \frac{X + X^2 + X^3}{1 - X^4} * \frac{1 + X + X^3 + X^4}{1 - X^5} * \frac{1 + X + X^2 + X^3 + X^5}{1 - X^6} \\ &= \frac{X + X^3 + X^4}{1 - X^5} * \frac{1 + X + X^2 + X^3 + X^5}{1 - X^6} \\ &= \frac{X}{1 - X^2} = \alpha_2^0. \end{aligned}$$

The work goes in the computation of the stream compositions, using the differential equation of Definition 5. This may be bothersome by hand but can easily be automated, using for instance a tool for coinductive proofs such as CIRC [12].

4. Featured example: Moessner's theorem for $k < 4$

As our featured example we use the periodic stream samplers to derive the first few cases of an elementary, yet elegant, result about integer powers known as Moessner's theorem.

Theorem 6 (*Moessner*). *Assume $k > 0$. Start from the infinite sequence of ones:* $(1, 1, \ldots)$*, drop every k+1st element, and form the sequence of partial sums of the resulting subsequence. Next drop every kth element and form the sequence of partial sums again. Next drop every $k-1$st element and form the sequence of partial sums again. Repeating this procedure k times one obtains the sequence of the kth powers of the integers.*

For $k = 2$ applying the procedure of this theorem results in the following streams.

$(1, 1, 1, 1, \ldots)$	
$(1, 1, 1, 1, \ldots)$	every 3rd element of the above stream dropped,
$(1, 2, 3, 4, \ldots)$	partial sums of the above stream,
$(1, 3, 5, 7, \ldots)$	every 2nd element of the above stream dropped,
$(1, 4, 9, 16, \ldots)$	partial sums of the above stream.

The theorem is posed in [14] and is proven by induction in [15]. More recently it has been proven using stream functions by Hinze [8]. Here we tackle it using stream samplers and prove some base cases using coinduction.

First we need some additional constants and functions, which we define as follows.

$\Omega(n) = n + 1$	(the stream $(1, 2, 3, \ldots)$),
$(k \cdot \sigma)(n) := k \cdot (\sigma(n))$	(scalar multiplication),
$(\sigma^{<k>})(n) := \sigma(n)^k$	(Hadamard exponentiation),
$(\Sigma\sigma)(n) = \sum_{i=0}^{n} \sigma(i)$	(partial sums).

Note that in the scalar multiplication and exponentiation $\cdot$ and power on the right hand side denote the multiplication and exponentiation on $\mathbb{N}$.

Here and in Section 2 we define the stream functions and constants by giving their n-th value. In coinductive proofs however, when simplifying stream equations it is easier to work with stream differential equations. Hence we state the following result that captures the stream differential equation for the operations that we need.

Proposition 7. *Let $j \in \mathbb{N}$ and $\sigma, \tau \in \mathbb{N}^\omega$. The stream operations $\overline{\ }$, $+$, $\_^k$, $k\cdot$ and Σ and the stream constant Ω satisfy the following stream differential equations.*

$\overline{k}(0) = k$,	$(\overline{k})' = \overline{k}$,
$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$,	$(\sigma + \tau)' = \sigma' + \tau'$,
$\Omega(0) = 1$,	$\Omega' = \Omega + \overline{1}$,
$(k \cdot \sigma)(0) = k\sigma(0)$,	$(k \cdot \sigma)' = k \cdot \sigma'$,
$\sigma^{<k>}(0) = \sigma(0)^k$,	$(\sigma^{<k>})' = (\sigma')^{<k>}$,
$\Sigma\sigma(0) = \sigma(0)$,	$(\Sigma\sigma)' = \Sigma\sigma' + \overline{\sigma(0)}$.

Proof. While there are several ways to prove these properties, including those using coinduction [16], here we give a direct calculational proof for the last equation. The equations for other functions can be derived in a similar manner. For the last equation note that given $n \in \mathbb{N}$ we have

$$(\Sigma\sigma)'(n) = (\Sigma\sigma)(n+1) = \sum_{i=0}^{n+1} \sigma(i) = \sum_{i=1}^{n+1} \sigma(i) + \sigma(0)$$
$$= \sum_{i=0}^{n} \sigma'(i) + \sigma(0) = (\Sigma\sigma')(n) + \overline{\sigma(0)}(n) = (\Sigma\sigma' + \overline{\sigma(0)})(n)$$

where the last step follows from the definition of $+$ in Section 2. □

Definition 8. We denote $\Sigma \circ D^i_k$ by Σ^i_k. Using Definition 4 we can derive the following equations written in terms of this notation.

$$\Sigma^i_k\sigma(0) = \begin{cases} \sigma(1) & i = 0, \\ \sigma(0) & i > 0; \end{cases} \qquad (\Sigma^i_k\sigma)' = \begin{cases} \Sigma^{k-2}_k(\sigma'') + \overline{\sigma(1)} & i = 0, \\ \Sigma^{i-1}_k(\sigma') + \overline{\sigma(0)} & i > 0. \end{cases}$$

As a final tool we need some properties regarding the composition of the above functions.

Proposition 9. *Assume* $\sigma, \tau \in \mathbb{N}^\omega$, $k \geq 2$ *and* $i < k$, *then*

(i) *for all* $m, n \in \mathbb{N}$, $\overline{m} + \overline{n} = \overline{m+n}$;
(ii) *for all* $m, n \in \mathbb{N}$, $(m+n)\cdot\sigma = m\cdot\sigma + n\cdot\sigma$;
(iii) $D^i_k(\overline{1}) = \overline{1}$;
(iv) $D^i_k(\sigma + \tau) = D^i_k(\sigma) + D^i_k(\tau)$;
(v) $\Sigma(\sigma + \tau) = \Sigma\sigma + \Sigma\tau$;
(vi) $\Sigma^i_k(\sigma + \tau) = \Sigma^i_k\sigma + \Sigma^i_k\tau$;
(vii) $(\sigma + \overline{1})^{<2>} = \sigma^{<2>} + 2\cdot\sigma + \overline{1}$;
(viii) $(\sigma + \overline{1})^{<3>} = \sigma^{<3>} + 3\cdot\sigma^{<2>} + 3\cdot\sigma + \overline{1}$;

Proof. Property (vi) follows from (iv) and (v). All the other properties are provable by straightforward coinduction. To demonstrate this point we prove (iv). Consider the relation

$$R = \{\langle D^i_k(\sigma + \tau), D^i_k(\sigma) + D^i_k(\tau)\rangle \mid \sigma, \tau \in \mathbb{N}^\omega \wedge i, k \in \mathbb{N} \wedge i < k\}$$

and assume $\langle \alpha, \beta\rangle \in R$. Then $\langle \alpha, \beta\rangle$ is necessarily of the form $\langle D^i_k(\sigma + \tau), D^i_k(\sigma) + D^i_k(\tau)\rangle$ for some σ, τ, i, k. First assume $i = 0$. We calculate:

$$\begin{aligned}\alpha(0) &= D^0_k(\sigma + \tau)(0) = (\sigma + \tau)(1) = \sigma(1) + \tau(1)\\ &= D^0_k(\sigma)(0) + D^0_k(\tau)(0) = \big(D^0_k(\sigma) + D^0_k(\tau)\big)(0) = \beta(0)\end{aligned}$$

and

$$\begin{aligned}\langle \alpha', \beta'\rangle &= \langle \big(D^0_k(\sigma + \tau)\big)', \big(D^0_k(\sigma) + D^0_k(\tau)\big)'\rangle\\ &= \langle D^{k-2}_k\big((\sigma + \tau)''\big), \big(D^0_k(\sigma)\big)' + \big(D^0_k(\tau)\big)'\rangle\\ &= \langle D^{k-2}_k(\sigma'' + \tau''), D^{k-2}_k(\sigma'') + D^{k-2}_k(\tau'')\rangle \in R;\end{aligned}$$

because $0 \leq k-2 < k$ and $\sigma'', \tau'' \in \mathbb{N}^\omega$. Next assume $i > 0$. We have

$$\begin{aligned}\alpha(0) &= D^i_k(\sigma + \tau)(0) = (\sigma + \tau)(0) = \sigma(0) + \tau(0)\\ &= D^i_k(\sigma)(0) + D^i_k(\tau)(0) = \big(D^i_k(\sigma) + D^i_k(\tau)\big)(0) = \beta(0)\end{aligned}$$

and

$$\begin{aligned}\langle \alpha', \beta'\rangle &= \langle \big(D^i_k(\sigma + \tau)\big)', \big(D^i_k(\sigma) + D^i_k(\tau)\big)'\rangle\\ &= \langle D^{i-1}_k\big((\sigma + \tau)'\big), \big(D^i_k(\sigma)\big)' + \big(D^i_k(\tau)\big)'\rangle\\ &= \langle D^{i-1}_k(\sigma' + \tau'), D^{i-1}_k(\sigma') + D^{i-1}_k(\tau')\rangle \in R;\end{aligned}$$

because $0 \leq i-1 < k$ and $\sigma', \tau' \in \mathbb{N}^\omega$. Therefore R is a stream bisimulation, and (iv) holds by coinduction proof principle. □

We are ready to prove Moessner's theorem for $k < 4$. We present two different proofs in our framework. First we prove the theorem by explicitly building stream bisimulation relations and using the coinduction proof principle (Sections 4.1–4.3). In the second proof we use the stream calculus and present a proof by calculating rational expressions (Section 4.4). We remark that both of these techniques can be extended to give proofs of Moessner's theorem for arbitrary k, but as our purpose here is to show the applications of our framework this generalisation falls out of the scope of the present work.

4.1. Degenerate case: stream of naturals

Define the relation $R_0 \subseteq \mathbb{N}^\omega \times \mathbb{N}^\omega$ inductively, to be the *smallest* relation satisfying the following properties.

(I0) $\forall \sigma \in \mathbb{N}^\omega, \langle \sigma, \sigma\rangle \in R_0$;
(II0) $\forall \sigma_0\sigma_1\tau_0\tau_1 \in \mathbb{N}^\omega, \langle \sigma_0, \tau_0\rangle \in R_0 \wedge \langle \sigma_1, \tau_1\rangle \in R_0 \implies \langle \sigma_0 + \sigma_1, \tau_0 + \tau_1\rangle \in R_0$;
(III0) $\langle \Sigma^0_2(\overline{1}), \Omega\rangle \in R_0$;
(IV0) $\langle \Sigma^1_2(\overline{1}), \Omega\rangle \in R_0$.

(We note that the last pair could have been omitted, as it is equal to the one but last pair. We have included it because in this way, the definition of R_0 above and those of R_1 and R_2 below all have the same structure.)

Theorem 10. R_0 *is a stream bisimulation.*

Proof. The proof is given in Appendix. □

Corollary 11 (*Moessner's Theorem for* $k = 1$)**.** $\Sigma^1_2(\overline{1}) = \Omega$.

4.2. Base case: stream of squares

Define the relation $R_1 \subseteq \mathbb{N}^\omega \times \mathbb{N}^\omega$ inductively, to be the *smallest* relation satisfying the following properties.

(I1) $\forall \sigma \in \mathbb{N}^\omega, \langle \sigma, \sigma \rangle \in R_1$;
(II1) $\forall \sigma_0\sigma_1\tau_0\tau_1 \in \mathbb{N}^\omega, \langle \sigma_0, \tau_0 \rangle \in R_1 \wedge \langle \sigma_1, \tau_1 \rangle \in R_1 \implies \langle \sigma_0 + \sigma_1, \tau_0 + \tau_1 \rangle \in R_1$;
(III1) $\langle \Sigma_2^0 \Sigma_3^1(\overline{1}), \Omega^{<2>} + \Omega \rangle \in R_1$;
(IV1) $\langle \Sigma_2^1 \Sigma_3^2(\overline{1}), \Omega^{<2>} \rangle \in R_1$.

Theorem 12. *R_1 is a stream bisimulation.*

Proof. The proof is given in Appendix. □

Corollary 13 (*Moessner's Theorem for $k = 2$*)**.** $\Sigma_2^1 \Sigma_3^2(\overline{1}) = \Omega^{<2>}$.

4.3. Second case: stream of cubes

Define the relation $R_2 \subseteq \mathbb{N}^\omega \times \mathbb{N}^\omega$ inductively, to be the *smallest* relation satisfying the following properties.

(I2) $\forall \sigma \in \mathbb{N}^\omega, \langle \sigma, \sigma \rangle \in R_2$;
(II2) $\forall \sigma_0\sigma_1\tau_0\tau_1 \in \mathbb{N}^\omega, \langle \sigma_0, \tau_0 \rangle \in R_2 \wedge \langle \sigma_1, \tau_1 \rangle \in R_2 \implies \langle \sigma_0 + \sigma_1, \tau_0 + \tau_1 \rangle \in R_2$;
(III2) $\langle \Sigma_2^0 \Sigma_3^1 \Sigma_4^2(\overline{1}), \Omega^{<3>} + 2{\cdot}\Omega^{<2>} + \Omega \rangle \in R_2$;
(IV2) $\langle \Sigma_2^1 \Sigma_3^2 \Sigma_4^3(\overline{1}), \Omega^{<3>} \rangle \in R_2$.

Theorem 14. *R_2 is a stream bisimulation.*

Proof. Similar to that for R_1, see Appendix. □

Corollary 15 (*Moessner's Theorem for $k = 3$*)**.** $\Sigma_2^1 \Sigma_3^2 \Sigma_4^3(\overline{1}) = \Omega^{<3>}$.

4.4. Proof using the stream calculus

We can derive Corollaries 11, 13 and 15 above by directly calculating the rational expressions for both sides and using the fundamental theorem of stream calculus.

Let us define

$$P_k = \Sigma_2^1 \Sigma_3^2 \cdots \Sigma_{k+1}^k(\overline{1}),$$
$$Q_k = \Sigma_2^0 \Sigma_3^1 \cdots \Sigma_{k+1}^{k-1}(\overline{1}).$$

First note that by unfolding the definition of Σ_2^0 we have

$$Q_1' = Q_1 + \overline{1},$$

and hence by the fundamental theorem[2]

$$Q_1 = Q_1(0) + X \times Q_1' = 1 + X \times (Q_1 + \overline{1}),$$

whence we obtain

$$\begin{aligned} Q_1 &= \frac{1}{1-X} + \frac{X \times \overline{1}}{1-X} \\ &= \frac{1}{1-X} + \frac{X}{(1-X)^2} \\ &= \frac{1}{(1-X)^2}. \end{aligned} \tag{4.1}$$

Note that here we use the readily derivable equation (cf. [16, Example 5.3]):

$$\overline{1} = \frac{1}{1-X}.$$

Similarly we have

$$P_1' = Q_1 + \overline{1},$$

[2] Recall that in such expressions we write r for $[r]$.

which by using the fundamental theorem and substituting (4.1) gives us

$$P_1 = 1 + X \times (Q_1 + \overline{1}) = \frac{1}{(1-X)^2}.$$

(Note that $P_1 = Q_1$, as one should expect.)

Carrying on in a similar fashion, by a repeated unfolding of the definition of Q_2 and using Proposition 9 we have

$$Q_2' = Q_2 + 2 \cdot Q_1 + \overline{2},$$

and again by the fundamental theorem

$$Q_2 = Q_2(0) + X \times Q_2' = 2 + X \times (Q_2 + 2 \cdot Q_1 + \overline{2}).$$

Therefore,

$$Q_2 = \frac{2}{1-X} + \frac{X}{1-X}(2 \cdot Q_1 + \overline{2}). \tag{4.2}$$

Likewise,

$$P_2' = Q_2 + Q_1 + \overline{1},$$

which by using the fundamental theorem and subsequently substituting (4.1) and (4.2) gives us

$$P_2 = \frac{1+X}{(1-X)^3}.$$

Repeating the calculations for $n = 3$ in the style above we get

$$Q_3' = Q_3 + 3 \cdot Q_2 + 4 \cdot Q_1 + \overline{4},$$
$$Q_3 = \frac{4}{(1-X)} + \frac{X}{1-X}(3 \cdot Q_2 + 4 \cdot Q_1 + \overline{4}),$$
$$P_3' = Q_3 + Q_2 + Q_1 + \overline{1},$$

which, alongside (4.1) and (4.2), results in the following closed form rational expression for P_3.

$$P_3 = \frac{1 + 4X + X^2}{(1-X)^4}.$$

All that remains is to find expressions for the right hand side of the identities in the above corollaries. This can be done in a similar fashion by using the fundamental theorem and Propositions 7 and 9. One can thus recover Corollary 11 as follows.

$$\begin{aligned}
\Omega &= \Omega(0) + X \times \Omega' \\
&= 1 + X \times (\Omega + \overline{1}) \\
&= \frac{1}{1-X} + \frac{X \times \overline{1}}{1-X} \\
&= \frac{1}{(1-X)^2} \\
&= P_1.
\end{aligned}$$

Furthermore,

$$\begin{aligned}
\Omega^{<2>} &= \Omega^{<2>}(0) + X \times (\Omega^{<2>})' \\
&= 1 + X \times (\Omega + \overline{1})^{<2>} \\
&= 1 + X \times (\Omega^{<2>} + 2 \cdot \Omega + \overline{1}),
\end{aligned}$$

which implies Corollary 13 as follows.

$$\begin{aligned}
\Omega^{<2>} &= \frac{1}{1-X} + \frac{X}{1-X}(2 \cdot \Omega + \overline{1}) \\
&= \frac{1}{1-X} + \frac{X}{1-X}\left(\frac{2}{(1-X)^2} + \overline{1}\right) \\
&= \frac{(1+X)}{(1-X)^3} \\
&= P_2.
\end{aligned}$$

Finally

$$\begin{aligned}\Omega^{<3>} &= \Omega^{<3>}(0) + X \times (\Omega^{<3>})' \\ &= 1 + X \times (\Omega + \overline{1})^{<3>} \\ &= 1 + X \times (\Omega^{<3>} + 3 \cdot \Omega^{<2>} + 3 \cdot \Omega + \overline{1}),\end{aligned}$$

which, when combined with the above closed expressions for $\Omega^{<2>}$, implies Corollary 15 as follows.

$$\begin{aligned}\Omega^{<3>} &= \frac{1}{1-X} + \frac{X}{1-X}(3 \cdot \Omega^{<2>} + 3 \cdot \Omega + \overline{1}) \\ &= \frac{1}{1-X} + \frac{X}{1-X}\left(\frac{3(1+X)}{(1-X)^3} + \frac{3}{(1-X)^2} + \overline{1}\right) \\ &= \frac{(1+4X+X^2)}{(1-X)^4} \\ &= P_3.\end{aligned}$$

5. Splitting and merging

All periodic stream samplers and, more generally, many periodic stream transformers that not necessarily preserve the order of the elements in a stream, can be obtained by splitting and merging streams. In this section, we introduce the operators of take and zip, with which streams can be split and merged, and we present a few basic laws about them.

Definition 16. (i) For $l \geq 2$ and $0 \leq i < l$, the *take operator* $T_l^i : A^\omega \to A^\omega$ is defined by the following stream differential equation:

$$T_l^i(\sigma)' = T_l^i(\sigma^{(l)}), \qquad T_l^i(\sigma)(0) = \sigma(i).$$

(ii) For $k \geq 1$ and streams $\sigma_0, \ldots \sigma_{k-1} \in A^\omega$, the *zip operator* $Z_k : (A^\omega)^k \to A^\omega$ is defined by the stream differential equation

$$\begin{aligned}&Z_k(\sigma_0, \ldots, \sigma_{k-1})(0) = \sigma_0(0), \\ &Z_k(\sigma_0, \ldots, \sigma_{k-1})' = Z_k(\sigma_1, \ldots, \sigma_{k-1}, \sigma_0').\end{aligned}$$

(Note that $\sigma_0, \ldots, \sigma_{k-1}$ above are *streams*, not *elements* of streams, which for a stream σ we denote by $\sigma(0)$, $\sigma(1)$, etc.) Examples are

$$\begin{aligned}&T_3^2(\sigma) = (\sigma(2), \sigma(5), \sigma(8), \ldots), \\ &Z_2(\sigma, \tau) = (\sigma(0), \tau(0), \sigma(1), \tau(1), \sigma(2), \tau(2), \ldots).\end{aligned}$$

As suggested by the latter, it is easy to see (by induction) that in general if $0 \leq r \leq k-1$ then

$$Z_k(\sigma_0, \ldots, \sigma_{k-1})(kn + r) = \sigma_r(n). \tag{5.1}$$

Furthermore, the following result holds which is the analogous of Proposition 3. Despite its specific format, it is very useful in that it allows the reconstruction of arbitrary *zip* operators from those with a prime arity.

$$\begin{aligned}&Z_k\big(Z_l(\sigma_0, \sigma_k, \ldots, \sigma_{lk-k}), Z_l(\sigma_1, \sigma_{k+1}, \ldots, \sigma_{lk-k+1}), \ldots, \\ &Z_l(\sigma_{k-1}, \sigma_{2k-1}, \ldots, \sigma_{lk-1})\big) = Z_{lk}(\sigma_0, \sigma_1, \ldots, \sigma_{lk-1}).\end{aligned} \tag{5.2}$$

Any periodic stream sampler can be expressed in terms of take and zip. With S as in Definition 2, we have

$$S(\sigma) = Z_k(T_l^{n_0}(\sigma), T_l^{n_1}(\sigma), \ldots, T_l^{n_{k-1}}(\sigma)).$$

More generally, we can define with take and zip periodic stream transformers that not merely produce substreams but that can change also the order of the elements. For instance, we can define the operation $Rev_k : A^\omega \to A^\omega$ of stream *reverse*, for any $k \geq 1$, by

$$Rev_k(\sigma) = Z_k(T_k^{k-1}(\sigma), T_k^{k-2}(\sigma), \ldots, T_k^0(\sigma)).$$

For instance,

$$Rev_3(\sigma) = (\sigma(2), \sigma(1), \sigma(0), \sigma(5), \sigma(4), \sigma(3), \ldots).$$

Next we present a few basic laws for take and zip that will allow us to prove elementary properties on stream transformers by equational reasoning. All of the identities below can easily be proved by coinduction.

Proposition 17. *For all $k \geq 1$, $l \geq 2$, $0 \leq i < l$,*

$$Z_k(T_k^0(\sigma), \ldots, T_k^{k-1}(\sigma)) = \sigma,$$
$$T_l^i(Z_l(\sigma_0, \ldots, \sigma_{l-1})) = \sigma_i,$$
$$T_l^i(\sigma) = Z_k(T_{k\times l}^i(\sigma), T_{k\times l}^{l+i}(\sigma), \ldots, T_{k\times l}^{(k-1)\times l+i}(\sigma)).$$

Let us illustrate these identities with an equational proof of our earlier example, the Drop expansion rule: for all $\sigma \in A^\omega$,

$$D_2^0(\sigma) = D_4^0 \circ D_5^2 \circ D_6^4(\sigma).$$

Let $\tau = D_6^4(\sigma)$. We have

$$\tau = D_6^4(\sigma) = Z_5(T_6^0(\sigma),\ T_6^1(\sigma),\ T_6^2(\sigma),\ T_6^3(\sigma),\ T_6^5(\sigma)).$$

Next let $\rho = D_5^2 \circ D_6^4(\sigma)$; it satisfies

$$\begin{aligned}\rho &= D_5^2(\tau) = Z_4(T_5^0(\tau),\ T_5^1(\tau),\ T_5^3(\tau),\ T_5^4(\tau))\\ &= Z_4(T_6^0(\sigma),\ T_6^1(\sigma),\ T_6^3(\sigma),\ T_6^5(\sigma)).\end{aligned}$$

Finally, we compute

$$\begin{aligned}D_4^0 \circ D_5^2 \circ D_6^4(\sigma) &= D_4^0(\rho) = Z_3(T_4^1(\rho),\ T_4^2(\rho),\ T_4^3(\rho))\\ &= Z_3(T_6^1(\sigma),\ T_6^3(\sigma),\ T_6^5(\sigma))\\ &= T_2^1(\sigma) = D_2^0(\sigma).\end{aligned}$$

As a second example, we prove $Rev_3 \circ Rev_3(\sigma) = \sigma$. Putting $\tau = Rev_3(\sigma)$,

$$\tau = Rev_3(\sigma) = Z_3(T_3^2(\sigma),\ T_3^1(\sigma),\ T_3^0(\sigma)).$$

It follows that

$$\begin{aligned}Rev_3 \circ Rev_3(\sigma) &= Rev_3(\tau) = Z_3(T_3^2(\tau),\ T_3^1(\tau),\ T_3^0(\tau))\\ &= Z_3(T_3^0(\sigma),\ T_3^1(\sigma),\ T_3^2(\sigma)) = \sigma.\end{aligned}$$

In the above, we have illustrated that the operators of take and zip are interesting because they can express all periodic stream samplers and because they can moreover be used to define stream transformers that have a periodic behaviour but that are not stream samplers. We have not given a general definition of periodic stream transformer. We shall come back to this point later.

6. Preserving rationality

In this section, we show that the result of applying the operators of take and zip to rational streams in A^ω is again rational. We shall use the following definition from [17, p.109].

Definition 18. For $\sigma \in A^\omega$ and $\rho \in A^\omega$ with $\rho(0) = 0$, we define the stream σ *applied to* ρ, written as $\sigma(\rho)$, by the following system of differential equations:

$$\sigma(\rho)' = \sigma'(\rho) \times \rho', \qquad \sigma(\rho)(0) = \sigma(0).$$

Recall from [17, Theorem 4.3] that every stream $\sigma \in A^\omega$ can be written as an infinite sum

$$\sigma = \sigma(0) + (\sigma(1) \times X) + (\sigma(2) \times X^2) + \ldots.$$

We may now think of $\sigma(\rho)$ as the stream that results from the above infinite sum by replacing every X by ρ (the condition $\rho(0) = 0$ will ensure that the resulting infinite sum is well-defined). In fact, the following result holds. Similar to the proof of [17, Theorem 4.3] this can be proven by coinduction.

Proposition 19. *For $\sigma, \rho \in A^\omega$ with $\rho(0) = 0$,*

$$\sigma(\rho) = \sigma(0) + (\sigma(1) \times \rho) + (\sigma(2) \times \rho^2) + \cdots.$$

This proposition reminds one of formal power series and (generating) function application (cf. [7]); note that the definition and identities above all live in stream calculus, where X is a constant stream and not a function variable.

If σ is polynomial and ρ is rational (with $\rho(0) = 0$) then $\sigma(\rho)$ is rational. Since for polynomials π and τ with $\tau(0) \neq 0$, one can easily show that

$$\frac{\pi}{\tau}(\rho) = \frac{\pi(\rho)}{\tau(\rho)},$$

it follows that if σ and ρ are rational then so is $\sigma(\rho)$. We shall be using the above mostly for the case that $\rho = X^n$, for some $n \geq 1$. For instance, we have

$$\frac{X}{(1-X)^2}(X^3) = \frac{X^3}{(1-X^3)^2}.$$

Since $X/(1-X)^2 = (0, 1, 2, \ldots)$ it follows that

$$\frac{X^3}{(1-X^3)^2} = (0, 0, 0, 1, 0, 0, 2, 0, 0, \ldots).$$

We are now ready to formulate our first preservation result. We remark that Corollary 21 and Proposition 23 below can be found in [3]. Our proofs are different: regarding *zip* the novelty lies in our use of coinduction proof principle in Proposition 20; regarding *take* we give a rather elementary proof in Proposition 23 while the proof in [3] is based on the Kleene–Schützenberger theorem.

Proposition 20. *Let $\sigma_0, \ldots, \sigma_{k-1} \in A^\omega$. Then*

$$Z_k(\sigma_0, \ldots, \sigma_{k-1}) = \sigma_0(X^k) + (X \times \sigma_1(X^k)) + \cdots + (X^{k-1} \times \sigma_{k-1}(X^k)).$$

Proof. Proof is by coinduction. □

Corollary 21. *The function* zip *preserves rationality: if $\sigma_0, \ldots, \sigma_{k-1} \in A^\omega$ are rational, for $k \geq 1$, then so is $Z_k(\sigma_0, \ldots, \sigma_{k-1})$.*

Next we show that the take operators preserve rationality as well. We shall use the following lemma; it has an easy proof by coinduction which we omit here.

Lemma 22. *Let $l \geq 2$ and $0 \leq i < l$.*

(a) *T_l^i is linear: for all $s, t \in A$, $\sigma, \tau \in A^\omega$,*

$$T_l^i((s \times \sigma) + (t \times \tau)) = (s \times T_l^i(\sigma)) + (t \times T_l^i(\tau)).$$

(b) *For $i > 0$ and $\sigma \in A^\omega$,*

$$T_l^i(X \times \sigma) = T_l^{i-1}(\sigma), \qquad T_l^0(X \times \sigma) = X \times T_l^{l-1}(\sigma).$$

Proposition 23. *The function* take *preserves rationality: if $\sigma \in A^\omega$ is rational then so is $T_l^i(\sigma)$, for all $l \geq 2$ and $0 \leq i < l$.*

Proof. By Lemma 22, it is sufficient to prove the proposition for streams of the form $1/\sigma$, with σ polynomial and $\sigma(0) \neq 0$. So let

$$\sigma = s_0 + s_1 X + \cdots + s_d X^d$$

be a polynomial stream, for $d \geq 0$ and $s_0, s_1, \ldots, s_d \in A$ with $s_0 \neq 0$. We claim that for any $l \geq 0$, the l-th stream derivative of $1/\sigma$ is of the form

$$(1/\sigma)^{(l)} = (r_0 + r_1 X + \cdots + r_{d-1} X^{d-1}) \times 1/\sigma \tag{6.1}$$

for certain $r_0, \ldots, r_{d-1} \in A$. For instance, we saw in Section 2 that

$$\left(\frac{1}{1-2X+X^2}\right)' = \left(\frac{1}{(1-X)^2}\right)' = \frac{2-X}{(1-X)^2}$$

Taking the second derivative yields

$$\begin{aligned}\left(\frac{1}{(1-X)^2}\right)^{(2)} &= \left(\frac{2-X}{(1-X)^2}\right)' \\ &= \left(\frac{2}{(1-X)^2}\right)' + \left(\frac{-X}{(1-X)^2}\right)' \\ &\quad [\text{using } (\sigma+\tau)' = \sigma' + \tau'] \\ &= \frac{4-2X}{(1-X)^2} + \frac{-1}{(1-X)^2} \\ &\quad [\text{using } (2\times\sigma)' = 2\times(\sigma') \text{ and identity (2.3)}] \\ &= \frac{3-2X}{(1-X)^2}.\end{aligned}$$

By induction, we can show that

$$\left(\frac{1}{(1-X)^2}\right)^{(l)} = \frac{(l+1)-lX}{(1-X)^2}.$$

The general case can be proved similarly (see [17] for the details).

Now for $l \geq 2$ and $0 \leq i < l$, we have

$$\begin{aligned}T_l^i(1/\sigma)' &= T_l^i((1/\sigma)^{(l)}) \quad [\text{by definition}] \\ &= T_l^i((r_0 + \cdots + r_{d-1}X^{d-1}) \times 1/\sigma) \quad [\text{by identity (6.1) above}] \\ &= (\rho_0 \times T_l^0(1/\sigma)) + \cdots + (\rho_{l-1} \times T_l^{l-1}(1/\sigma))\end{aligned}$$

for certain rational streams $\rho_0, \ldots, \rho_{l-1} \in A^\omega$, where the last equality follows from Lemma 22. Multiplying the equation by X and adding $(1/\sigma)(i)$ to both sides gives

$$\begin{aligned}T_l^i(1/\sigma) &= T_l^i(1/\sigma)(0) + (X \times T_l^i(1/\sigma)') \quad [\text{by the fundamental theorem, Section 2}] \\ &= (1/\sigma)(i) + \big(X \times \big((\rho_0 \times T_l^0(1/\sigma)) + \cdots + (\rho_{l-1} \times T_l^{l-1}(1/\sigma))\big)\big) \\ &= (1/\sigma)(i) + (X \times \rho_0 \times T_l^0(1/\sigma)) + \cdots + (X \times \rho_{l-1} \times T_l^{l-1}(1/\sigma)).\end{aligned}$$

We have an equation of this form for all i with $0 \leq i < l$. Thus we have obtained a system of l equations in l unknowns: $T_l^0(1/\sigma), \ldots, T_l^{l-1}(1/\sigma)$, where all the occurrences of the unknowns on the right are multiplied by a rational stream of the form $X \times \rho$. Such a system of what could be called *guarded* equations can easily be seen to have rational streams as solutions, essentially by standard linear algebraic reasoning. □

Corollary 24. *If an operator is built by function composition from: constant streams $[r]$ (for $r \in A$), X, sum $+$, convolution product $\times$, convolution inverse $(-)^{-1}$, and the zip and take operators Z_k and T_l^i, then it preserves rationality.*

Proof. For the constants, sum, product and inverse, this is trivial and for zip and take, we have Corollary 21 and Proposition 23. □

Here are some examples. Let $\sigma = 1/(1-X)^2 = (1, 2, 3, \ldots)$. We will compute

$$\alpha = T_3^0(\sigma), \qquad \beta = T_3^1(\sigma), \qquad \gamma = T_3^2(\sigma).$$

In the computation below, we shall be using the following equalities:

$$\sigma^{(3)} = \frac{4-3X}{(1-X)^2},$$

$$T_3^0(X\times\sigma) = X\times\gamma, \qquad T_3^1(X\times\sigma) = \alpha, \qquad T_3^2(X\times\sigma) = \beta.$$

For α, we compute as follows:

$$\alpha' = T_3^0(\sigma^{(3)}) = T_3^0\left(\frac{4-3X}{(1-X)^2}\right) = 4\alpha - (3X\times\gamma).$$

Using the fundamental theorem and $\alpha(0) = 1$ gives

$$\alpha = 1 + (4X\times\alpha) - (3X^2\times\gamma).$$

Similar computations lead to equations for β and γ:

$$\beta = 2 + (4X\times\beta) - (3X\times\alpha),$$

$$\gamma = 3 + (4X\times\gamma) - (3X\times\beta).$$

Solving this system of three equations gives

$$\alpha = \frac{1+2X}{(1-X)^2}, \qquad \beta = \frac{2+X}{(1-X)^2}, \qquad \gamma = \frac{3}{(1-X)^2}.$$

As a next example, we will compute $Rev_3(\sigma)$, as follows:

$$\begin{aligned} Rev_3(\sigma) &= Z_3(T_3^2(\sigma),\ T_3^1(\sigma),\ T_3^0(\sigma)) \quad \text{[definition } Rev_3\text{]} \\ &= Z_3\left(\frac{3}{(1-X)^2},\ \frac{2+X}{(1-X)^2},\ \frac{1+2X}{(1-X)^2}\right) \\ &= \frac{3}{(1-X^3)^2} + X \times \frac{2+X^3}{(1-X^3)^2} + X^2 \times \frac{1+2X^3}{(1-X^3)^2} \quad \text{[Proposition 20]} \\ &= \frac{3-X-X^2+2X^3}{(1-X)^2(1+X+X^2)}. \end{aligned}$$

7. Preserving algebraicity

Corollary 24 shows that starting with a rational stream and applying some 'basic' operations we stay in the realm of rational streams. But there is a somewhat larger class of streams that is preserved under some of these operations, namely the class of algebraic streams defined below.

Algebraicity is a notion that should be defined over other algebraic structures. In this section we study algebraicity over finite fields. For $q \geq 1$ let $\mathbb{F}_q$ be the finite field with q elements (note that $\mathbb{F}_q$ has cardinality p^n for some prime p [9]). A univariate polynomial in X is a polynomial of the form $a_0 + a_1X + \cdots + a_kX^k$ where $a_i \in \mathbb{F}_q$, $a_k \neq 0$. Subsequently by $\mathbb{F}_q(X)$ we denote the *field of fractions of polynomials in X*, i.e., $\pi(X) \in \mathbb{F}_q(X)$ means there are univariate polynomials $\pi_1(X)$, $\pi_2(X)$ with coefficients in $\mathbb{F}_q$ such that $\pi(X) = \pi_1(X)/\pi_2(X)$.

Definition 25. A stream $\sigma \in \mathbb{F}_q^\omega$ is *algebraic* over $\mathbb{F}_q(X)$ if there are $A_i \in \mathbb{F}_q(X)$, $A_k \neq 0$ such that $A_0 + A_1\sigma + \cdots + A_k\sigma^k = 0$.[3]

As an example, the stream $\sigma \in \mathbb{F}_2^\omega$ for which

$$X^3 + \frac{1}{1-X}\sigma + \frac{X+1}{1-X^2}\sigma^2 = 0,$$

is algebraic over $\mathbb{F}_2(X)$.

This definition is borrowed from the theory of formal power series [6] and is motivated by the fact that σ can be considered as the sequence of coefficients of a formal power series. Following Section 2, by taking $A := \mathbb{F}_q$ we can obtain the stream calculus on $\mathbb{F}_q^\omega$. As a consequence the left hand side of the expression above can be interpreted in two ways: as a stream in the stream calculus where $X = (0, 1, 0, \ldots)$ as in Section 2 or as a formal power series in the ring of formal power series with one variable X. It can easily be observed that each rational stream in $\mathbb{F}_q^\omega$ is algebraic. The converse does not always hold. In next section we give an example of an algebraic stream that is not rational, namely the *Prouhet–Thue–Morse* sequence. There are also streams that are not algebraic, a simple example being the Fibonacci sequence [6, Section 1.2.2]. But in general, the so called *automatic* streams, i.e., streams that are 'computable' by a class of transducers similar to Mealy machines,[4] can be shown to be algebraic [6].

We state a useful criterion, originally from [5], that is usually used as an intermediate step in relating algebraic and automatic streams but here we will use it on its own. Our formulation follows [6, Theorem 3.2.1].

Definition 26. Let $\sigma \in \mathbb{F}_q^\omega$. Then the *q-kernel* of σ is the set of substreams of σ defined as

$$N_q(\sigma) = \{\lambda n.\sigma(q^s n + r) \mid s \geq 0,\ 0 \leq r \leq q^s - 1\}. \tag{7.1}$$

Here $\lambda n.f(n)$ is the notation for the stream whose nth element is $f(n)$.

For a very simple example, let $q = 2$ and $\sigma = (1, 2, 1, 2, 1, 2, \ldots)$. Then

$$N_q(\sigma) = \{(1, 2, 1, 2, 1, 2, \ldots),\ (1, 1, 1, \ldots),\ (2, 2, 2, \ldots)\}$$

Theorem 27 (*Christol*)**.** *A stream $\sigma \in \mathbb{F}_q^\omega$ is algebraic over $\mathbb{F}_q(X)$ if and only if the q-kernel $N_q(\sigma)$ of σ is finite.*

By applying this theorem we can obtain what can be considered as counterparts of Corollary 21 and Proposition 23 above. First, we have the following which resembles Proposition 23. This one is an easy consequence and is also mentioned in [6, Section 3.2.2], so we skip the proof.

[3] In fact we can restrict the coefficients A_i to *univariate polynomials* instead of fractions.

[4] This is a very informal description. The precise definition of automatic sequences can be found in [1].

Proposition 28. *The function* take *preserves algebraicity for streams over a finite alphabet: if $\sigma \in \mathbb{F}_q^\omega$ is algebraic over $\mathbb{F}_q(X)$ then so is $T_l^i(\sigma)$, for all $l \geq 2$ and $0 \leq i < l$.*

For *zip* we first need to define a notion based on q-kernels.

Definition 29. Let $\sigma_0, \ldots, \sigma_{h-1} \in \mathbb{F}_q^\omega$ (where $h > 0$). Then the *h-fold q-kernel* of $\sigma_0, \ldots, \sigma_{h-1}$ is the set of sequences defined as

$$N_q^{(h)}(\sigma_0, \ldots, \sigma_{h-1}) = \{Z_h(\tau_0, \ldots, \tau_{h-1}) \mid \forall i\, \exists j,\ \ \tau_i \in N_q(\sigma_j)\}. \tag{7.2}$$

Note that we have the following trivial properties.

Proposition 30. (i) *Let $\varsigma_0, \ldots, \varsigma_{h-1}$ be a possibly repetitive sequence of elements of the set $\{\sigma_0, \ldots, \sigma_{h-1}\}$. Then $N_q^{(h)}(\varsigma_0, \ldots, \varsigma_{h-1}) \subseteq N_q^{(h)}(\sigma_0, \ldots, \sigma_{h-1})$.*
(ii) *If the q-kernel of each of $\sigma_0, \ldots, \sigma_{h-1}$ is finite then the h-fold q-kernel of them is finite.*

We use these facts for proving that *zip* preserves algebraicity. We remark that this result can also be derived from [1, Theorem 6.8.2] using the relationship between algebraic and automatic sequences. However, our proof is different in that it does not depend on the results on automatic sequences.

Proposition 31. *The function* zip *preserves algebraicity for streams over a finite alphabet: if $\sigma_0, \ldots, \sigma_{h-1} \in \mathbb{F}_q^\omega$ (where $h > 0$) are algebraic over $\mathbb{F}_q(X)$, then so is $Z_h(\sigma_0, \ldots, \sigma_{h-1})$.*

Proof. Let $\tau := Z_h(\sigma_0, \ldots, \sigma_{h-1})$. We show that

$$N_q(\tau) \subset N_q^{(h)}(\sigma_0, \ldots, \sigma_{h-1}). \tag{7.3}$$

The result then will follow from Theorem 27, since the right hand side is finite.

To prove (7.3) assume $\alpha \in N_q(\tau)$. Then $\alpha = \lambda n.\tau(q^s n + r)$ for some s, r as in (7.1). Furthermore by applying (5.1) it can easily be seen that

$$\alpha = Z_h(\lambda n.\tau(hnq^s + r), \lambda n.\tau((hn+1)q^s + r), \ldots, \lambda n.\tau((hn + (h-1))q^s + r)).$$

So α is the *zip* of h streams each of which is of the form $\tau((hn + k)q^s + r)$ where $k \leq h-1$. Assume, using the division algorithm, that $q = d_0 h + r_0$, $r = d_1 h + r_1$ and $kr_0^s + r_1 = d_k h + r_k$. Then

$$\begin{aligned}(hn+k)q^s + r &= hnq^s + k(d_0 h + r_0)^s + d_1 h + r_1 \\ &= hnq^s + k(d_0^s h^s + sd_0^{s-1}h^{s-1}r_0 + \cdots + sd_0 h r_0^{s-1} + r_0^s) + d_1 h + r_1 \\ &= h(nq^s + kd_0^s h^{s-1} + skd_0^{s-1}h^{s-2}r_0 + \cdots + skd_0 r^{s-1} + d_1) + d_k h + r_k \\ &= h(nq^s + U_k) + r_k,\end{aligned}$$

where

$$U_k = kd_0^s h^{s-1} + skd_0^{s-1}h^{s-2}r_0 + \cdots + skd_0 r^{s-1} + d_1 + d_k.$$

From this and using the property of *zip* in (5.1) we get

$$\lambda n.\tau((hn+k)q^s + r) = \lambda n.\tau(h(nq^s + U_k) + r_k) = \lambda n.\sigma_{r_k}(nq^s + U_k).$$

It remains to be checked whether $U_k < q^s$. But this is evident because

$$\begin{aligned}hU_k &= k(q^s - r_0^s) + d_1 h + d_k h \\ &= kq^s + r - r_k \\ &\leq (h-1)q^s + r \\ &< hq^s.\end{aligned}$$

Therefore defining $\upsilon_k := \lambda n.\sigma_{r_k}(nq^s + U)$ we obtain

$$\upsilon_0 \in N_q(\sigma_{r_0}), \ldots, \upsilon_{h-1} \in N_q(\sigma_{r_{h-1}})$$

such that

$$\alpha = Z_h(\upsilon_0, \ldots, \upsilon_{h-1}).$$

Hence, by (7.2) and Proposition 30 we have $\alpha \in N_q^{(h)}(\sigma_0, \ldots, \sigma_{h-1})$. □

In conclusion of this section, we remark that the sum and the Hadamard product of two algebraic streams are algebraic. The proof is a straightforward application of Theorem 27, together with a similar construct to the one in (7.2).

Proposition 32. *Binary operations of sum and the Hadamard product preserve algebraicity for streams over a finite alphabet: if $\sigma, \tau \in \mathbb{F}_q^\omega$ are algebraic over $\mathbb{F}_q(X)$, then so is $\sigma \star \tau$, where $\star \in \{+, \odot\}$.*

Proof. For $\star \in \{+, \odot\}$, define

$$N_q^\star(\sigma, \tau) = \{\theta_0 \star \theta 1 \mid \theta_0 \in N_q(\sigma) \wedge \theta_1 \in N_q(\tau)\}.$$

It suffices to show that $N_q(\sigma \star \tau) \subset N_q^\star(\sigma, \tau)$. To this end, assume $\alpha \in N_q(\sigma \star \tau)$. Then for some s, r as in (7.1) we have

$$\begin{aligned}\alpha &= \lambda n.(\sigma \star \tau)(q^s n + r)\\ &= \lambda n.\big(\sigma(q^s n + r) \star \tau(q^s n + r)\big)\\ &= \lambda n.\sigma(q^s n + r) \ \star \ \lambda n.\tau(q^s n + r).\end{aligned}$$

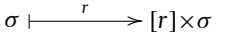
I.e., $\alpha \in N_q^\star(\sigma, \tau)$ by taking $\theta_0 := \lambda n.\sigma(q^s n + r)$ and $\theta_1 := \lambda n.\tau(q^s n + r)$. □

The above proof relies on the fact that the (stream) sum and the Hadamard product are element-wise extensions of sum and product from $\mathbb{F}_q$ to $\mathbb{F}_q^\omega$, i.e., $(\sigma \star \tau)(n) = \sigma(n) \star \tau(n)$ for each n. Evidently, this result can be generalised for any binary operation on streams that operates in such an element-wise manner.

8. Stream circuits

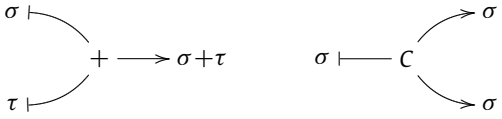
We briefly recall the correspondence between rational streams (of real numbers) and so-called stream circuits built from adder, copier, register and multiplier gates. Then we propose to look at stream circuits built from this set of gates extended with basic gates for splitting and merging. We study their behaviour by describing how they act on input streams of real numbers. For circuits without feedback, it will be immediate that they preserve rationality. For feedback circuits, the situation turns out to be more complicated.

Stream circuits [18] are data flow networks that act on streams of inputs (here real numbers) and produce streams of outputs. They are built out of four types of basic gates by means of composition, which amounts simply to connecting (single) output ends to (single) input ends. Below we describe the basic gates and their input–output behaviour. An *r-multiplier*, for $r \in A$, transforms an input stream $\sigma \in A^\omega$ into $[r]\times\sigma$:

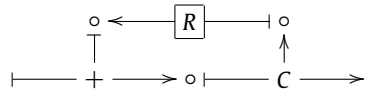


which amounts to the element-wise multiplication of the input values with r. A *register* (with initial value 0) takes an input stream σ

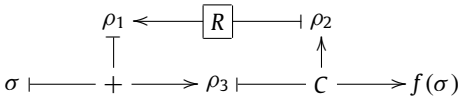
and outputs it with one step delay, after having output the initial value 0 first. An *adder* takes two input streams σ and τ and outputs the stream consisting of their element-wise addition, and a *copier* simply copies input streams into output streams:



Stream circuits are then built by composing various basic gates. Here is a simple example of a circuit with feedback:



For an input stream $\sigma \in A^\omega$, we can compute the output stream as a function $f(\sigma)$ of σ as follows. With the three internal composition nodes of the circuit, we associate streams $\rho_1, \rho_2, \rho_3 \in A^\omega$:



For each of the three basic gates used in this circuit, we have an equation:

$$\rho_1 = X \times \rho_2, \qquad \rho_3 = \sigma + \rho_1, \qquad \rho_2 = \rho_3 = f(\sigma).$$

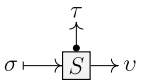
Eliminating the streams ρ_1, ρ_2 and ρ_3 from this system of equations, we find

$$f(\sigma) = \frac{1}{1 - X} \times \sigma.$$

In [18, Theorem 4.25], it is shown that every (finite) circuit possibly with feedback loops (which always have to pass through at least one register), compute stream functions $f : A^\omega \to A^\omega$ of the form: $f(\sigma) = \rho \times \sigma$, for all σ and some fixed rational stream ρ; conversely, every such function is implemented by some finite circuit.

Next we introduce new basic gates for the splitting and merging of streams.

A *splitter* gate in our setting is a gate with one input and two output ends:



It transforms an input stream $\sigma \in \mathbb{R}^\omega$ to streams τ, υ such that

$$\tau = D_2^1(\sigma) = T_2^0(\sigma), \qquad \upsilon = D_2^0(\sigma) = T_2^1(\sigma). \tag{8.1}$$

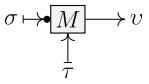
Note that $\tau = even(\sigma)$ and $\upsilon = even(\sigma')$ (where *even* is defined in Section 3). We define

$$odd(\sigma) := even(\sigma').$$

Hence the splitters transform σ to $even(\sigma)$ and $odd(\sigma)$.

The splitter is different from the previous ports (in particular copier) in that only one of its outgoing ports is active at any time. This means when a data element belonging to τ is being output, the port outputting υ is pending. Moreover, the active output port alternates with each data consumed from σ. The bullet on one of the output ports denotes the port that is activated in the very beginning. This confirms the fact that $\tau = even(\sigma)$.

A *merger* gate is a gate with two inputs and one output end.



It transforms two input streams $\sigma, \tau \in \mathbb{R}^\omega$ to a stream υ such that

$$\upsilon = Z_2(\sigma, \tau).$$

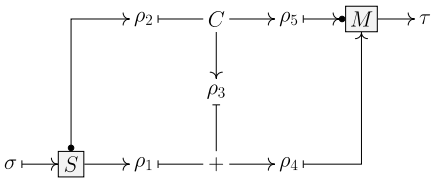
In contrast with the splitter gate, in a merger only one of the inputs is activated at a time. The active input port alternates with each data output. Again the bullet denotes the port that is activated in the very beginning, i.e., the one that contributes to $\upsilon(0)$.

It is clear that merger and splitter can be composed with each other and with the previously defined gates to form compound circuits. We call such a circuit an *extended stream circuit*. The functions $f(\sigma) = \rho \times \sigma$, for constant stream ρ, that are realisable by well-formed stream circuits are instances of *causal* functions on streams [18]. These are functions that output a data item after each input. Since each gate of stream circuit is causal their composition is causal too. However, introducing splitter and merger into the extended stream circuits leads to *overconsumption* (splitter) or *overproduction* (merger). So there will be data queues behind causal gates. Hence we need to assume the following important rule.

The connecting lines in extended stream circuits behave like unbounded FIFO buffers.

This is similar to the framework of Kahn Networks [10].

Simple feed-forward extended stream circuits can easily be analysed using the same method used for stream circuits. As an example consider the following circuit [13, Section 4].



First note that,

$$\rho_1 = odd(\sigma),$$
$$\rho_2 = \rho_3 = \rho_5 = even(\sigma),$$
$$\rho_4 = odd(\sigma) + even(\sigma),$$
$$\tau = Z_2(even(\sigma), odd(\sigma) + even(\sigma)).$$

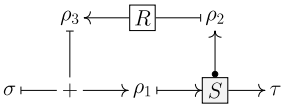
Assume that we input the stream $\sigma = X/(1-X)^2 = (0, 1, 2, \ldots)$ to the above circuit. It can easily be shown that (cf. the example at the end of Section 6),

$$even(\sigma) = \frac{2X}{(1-X)^2}, \qquad odd(\sigma) = \frac{1+X}{(1-X)^2}.$$

Subsequently we derive

$$\begin{aligned}\tau &= Z_2\left(\frac{2X}{(1-X)^2}, \frac{1+3X}{(1-X)^2}\right)\\ &= \frac{2X^2}{(1-X^2)^2} + \frac{X+3X^3}{(1-X^2)^2} = \frac{X(1+2X+3X^2)}{(1-X)^2(1+X)^2}.\end{aligned}$$

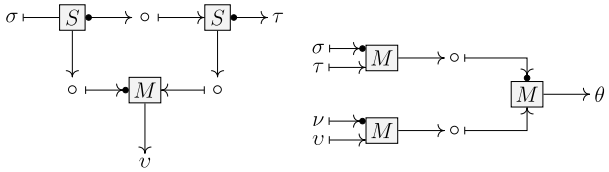
While feed-forward extended stream circuits are relatively easy to analyse, allowing feed-back will complicate the matter. First of all we need to formulate well-formedness rules with respect to the topology of the circuit, whose purpose would be to prevent overconsumption from happening (overproduction is not a problem, since we assume that connecting lines are buffers). Intuitively this means that for any possible path in the circuit, any chunk of subsequent splitters should be directly connected to the global input or be preceded by an appropriate number of mergers. In future work we plan to make such rules more formal. For now we give an example of a non-well-formed circuit demonstrating the problem of overconsumption.



In the circuit above, assuming there is a flow, one can take the second derivative of the behavioural equations for ρ_1 and obtain the contradiction in the form of following identity

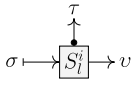
$$\rho_1(2) = \sigma(2) + \rho_1(2).$$

Next we consider the stream circuits for capturing n-ary mergers and splitters. Evidently, by sequencing binary splitters and mergers that we defined above, one can synthesise feed-forward circuits for calculating dyadic (2^n-ary) *take* and *zip* functions. I.e., we can build circuits for calculating $T^i_{2^n}$ and Z_{2^n}. For instance the circuits below calculate T^0_4 (left) and Z_4 (right). The result for *take* is a consequence of Proposition 3 (resp. Eq. (5.2) for *zip*).



One can readily verify that $\tau = T^0_4(\sigma)$ in the left-most circuit and $\theta = Z_4(\sigma, \nu, \tau, \upsilon)$ in the right-most circuit. The above construction suggests that by adding new splitter and merger gates with p input and output ports, where p is a prime number, we can synthesise circuits for calculating general *take* and *zip* functions T^l_n and Z_n. We do not consider this issue in the present paper. Instead, following [13] we consider another type of splitters and mergers, namely those with at most two inputs and outputs. Note that unlike the binary case (cf. (8.1) above), the behaviour of the *drop* function is not captured by the left-most circuit, i.e., $\upsilon \neq D^0_4(\sigma)$. In order to avoid such intricate 'loose ends' in compound circuits we opt for using Mak's splitter gates that capture both *take* and *drop*. Likewise, we use Mak's merger gates that indirectly allow us to calculate *zip* function. The behaviour in the general case is presented in the following definition.

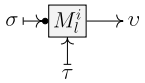
Definition 33 (*Mak [13]*)**.** Let $0 \leq i < l$. An l-ary splitter gate is a gate with one input and two outputs, depicted below,



that transforms σ to two streams τ and ν such that

$$\tau = T^i_l(\sigma), \qquad \nu = D^i_l(\sigma).$$

An l-ary merger gate is a gate with two inputs and one output, depicted below,

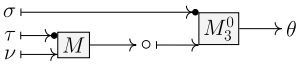


that transforms streams σ and τ into a stream ν such that

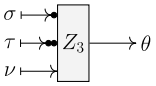
$$\sigma = T^i_l(\nu), \qquad \tau = D^i_l(\nu).$$

Note that in general we have l different l-ary splitters and mergers. According to the above definition the gates S and M described previously are the same as S_2^0 and M_2^0. It is clear that for $l = 2$, the two different possibilities for each of merger and splitter are isomorphic.

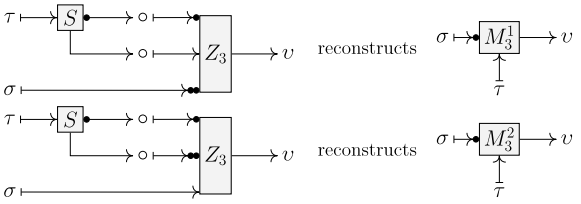
We emphasise that l-ary *zip* can be calculated by the sequencing of merger gates. The idea behind the general construction, which only uses merger gates of the form M_i^0, is manifested by the following circuit that calculates $\theta = Z_3(\sigma, \tau, \nu)$.



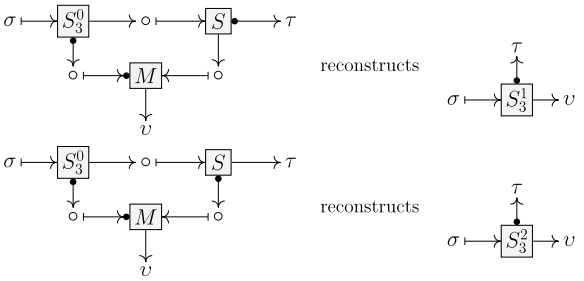
We will summarise the above circuit as



In general the merger gates $M_l^1, M_l^2, \ldots, M_l^{l-1}$ are redundant in that they can be constructed using M_l^0 and splitter and merger gates with arity lower than l. This result follows by induction on the arity. While we do not present this as a formal result, we note that the idea behind the general construction can be seen in the following reconstruction of M_3^1 and M_3^2.

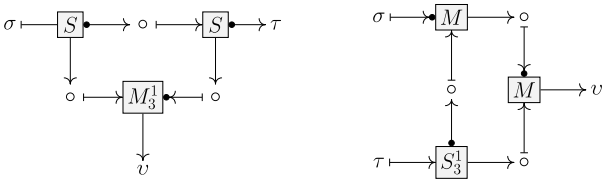


Hence the idea is to access the individual elements in each block size by sequencing appropriate number of splitter gates of the form S_i^0. As another consequence of this idea one can show that the splitter gates $S_l^1, S_l^2, \ldots, S_l^{l-1}$ are redundant. Again we demonstrate the idea behind the proof of this result by the following reconstruction of S_3^1 and S_3^2.

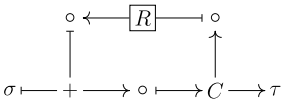


Summing up the above constructions we conclude that using the merger gates of the form M_i^0 and the splitter gates of the form S_i^0 (for $i > 1$) are sufficient for reconstructing all l-ary splitter and merger gates of Mak. Note that the above constructions are based on our assumption that the connecting lines are unbounded buffers.

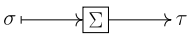
Coming back to the circuit for 4-ary samplers, we note that although S_4^0 can be used to obtain both *take* and *drop* samplers, we can reconstruct it using gates with lower arity. The circuits below reconstruct S_4^0 and M_4^0.



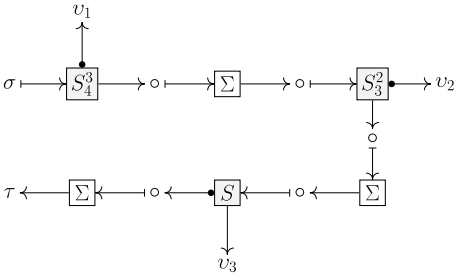
It is easy to build circuits that use the construction in Moessner's theorem from Section 4. First note that the partial sums operator is realised by the following feedback circuit, i.e., $\tau = \Sigma\sigma$.



If we summarise the above circuit as



we construct the circuit



which realises Moessner's construction for $k = 3$: if $\sigma = \overline{1} = (1, 1, 1, \ldots)$ is input to this circuit then

$$\tau = \Sigma_2^1 \Sigma_3^2 \Sigma_4^3(\overline{1}) = \Omega^3.$$

We conclude this section by giving an example of a non-rational stream that can be calculated using the extended stream circuits. This will demonstrate that adding splitter and merger will indeed extend the class of definable streams with respect to those of the ordinary stream calculus. Our example is the *Prouhet–Thue–Morse* sequence which is an algebraic non-rational[5] stream over $\mathbb{F}_2(X)$. The stream, which we denote by Ψ is given by the following behavioural differential equations.

$$\Psi(0) = 0, \qquad \Psi'(0) = 1,$$
$$\Psi'' = Z_2(\Psi', \widetilde{\Psi'});$$

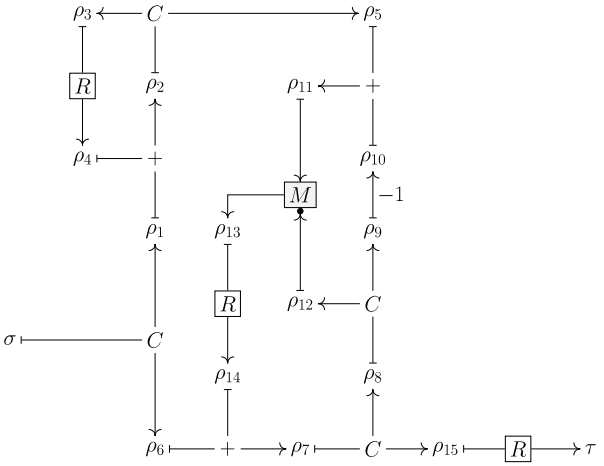
where $\widetilde{\sigma}$ is the bit-wise negation of σ itself defined as

$$\widetilde{\sigma}(0) = \neg\sigma(0), \qquad \widetilde{\sigma}' = \widetilde{\sigma'}.$$

Since we are working with bitstreams, i.e., since $\sigma \in \mathbb{F}_2^\omega$, we have

$$\widetilde{\sigma} = \overline{1} - \sigma = \frac{1}{1-X} - \sigma. \tag{8.2}$$

Consider the following extended circuit which contains only one merger.



Note that the -1-multiplier is meaningful since we are working in a field.

We calculate the intermediate values ρ_i as follows:

$$\rho_1 = \rho_6 = \sigma,$$
$$\rho_2 = \rho_3 = \rho_5 = \sigma + X \times \rho_2$$

which implies

$$\rho_2 = \frac{1}{1-X} \times \sigma,$$
$$\rho_4 = \frac{X}{1-X} \times \sigma.$$

[5] Proof of this fact can be found in [6].

Next we observe that

$$\begin{aligned}\rho_7 &= \rho_6 + \rho_{14}\\ &= \sigma + \rho_{14}.\end{aligned}$$

Furthermore,

$$\rho_8 = \rho_9 = \rho_{12} = \rho_{15} = \rho_7 = \sigma + \rho_{14}$$
$$\rho_{10} = -\rho_7$$

and

$$\begin{aligned}\rho_{11} &= \rho_5 + \rho_{10}\\ &= \frac{1}{1-X} \times \sigma - \rho_7.\end{aligned}$$

Also

$$\begin{aligned}\rho_{13} &= Z_2(\rho_{12}, \rho_{11})\\ &= Z_2\left(\rho_7, \frac{1}{1-X} \times \sigma - \rho_7\right).\end{aligned}$$

Similarly, we find

$$\rho_{14} = X \times Z_2\left(\rho_7, \frac{1}{1-X} \times \sigma - \rho_7\right),$$
$$\tau = X \times \rho_7.$$

From here we can obtain

$$\rho_7 = \sigma + X \times Z_2\left(\rho_7, \frac{1}{1-X} \times \sigma - \rho_7\right).$$

Hence if $\sigma = [1] = (1, 0, 0, \ldots)$ is input to this circuit then

$$\begin{aligned}\rho_7 &= 1 + X \times Z_2\left(\rho_7, \frac{1}{1-X} \times 1 - \rho_7\right)\\ &= 1 + X \times Z_2\left(\rho_7, \frac{1}{1-X} - \rho_7\right)\\ &= 1 + X \times Z_2(\rho_7, \widetilde{\rho_7}) \quad \text{[by (8.2)]}.\end{aligned}$$

Therefore ρ_7 satisfies the behavioural differential equation for Ψ', and thus $\rho_7 = \Psi'$ by the fundamental theorem. Subsequently

$$\tau = X \times \rho_7 = 0 + X \times \Psi' = \Psi.$$

9. Discussion and future work

We have studied various data independent operations for partitioning, projecting or merging streams. These operations are usually studied in the context of dataflow programming, while we showed that the operations and many of their properties can be defined using elements of *stream calculus*, namely behavioural differential equations for definitions and coinduction proof principle for proofs. Furthermore we focused on *take* and *zip* operations, for merging and splitting of data that are widely used elements in dataflow programming [4,13] and models of concurrency [2]. We dealt with the fact that splitting and merging preserves well behaved and well patterned class of streams namely rational and algebraic streams. While some of those results were known in the literature, we present them in the framework of stream calculus. Finally we showed how adding two new gates, namely dyadic merger and splitter will enlarge the class of streams that are realisable using stream circuits to beyond rational streams and into the realm of algebraic streams.

There are several issues and directions for future work.

Automated coinduction proofs. In Section 3 we showed how to use coinduction to prove the Drop exchange rule by finding a bisimulation. There are in fact tools for automatically finding bisimulation, e.g. the CIRC tool [12]. We applied CIRC and it could drive the rule $D_2^0 = D_4^0 \circ D_5^2 \circ D_6^4$. The CIRC tool uses a special technique called *circular coinduction*, a partial decision procedure, whose success depends on the type of bisimulation to be found. Our goal is to further investigate the different types of bisimulation that will arise in *Periodic Drop Take Calculus (PDTCS)* of Mak [13] and examine the applicability of circular coinduction to them.

Extended stream circuits. We plan to investigate precisely which class of streams are realisable using extended stream circuits of Section 8. For this we will further study the extended circuits with p-adic merger and splitter where p is a prime number. Moreover the question of well-formedness with respect to the topological properties of the circuits needs to be investigated. As a related problem we are interested in finding a closed formula for *even* and *odd* (and their n-ary counterparts). Intuitively these functions correspond to the roots of unity (cf. [22, Section 2.4], and Lemma 22 on periodicity of *take*). This implies that one could use hyperbolic functions (e.g. cosh) to represent the effect of *even* in the stream calculus. We plan to make this connection more formal.

Coalgebraic semantics. Earlier work on stream calculus has led to a coalgebraic treatment of rational power series [20]. An advantage of the coalgebraic modelling is that it present a unified way for dealing with stream circuits, stream functions and transducers. Above all it helps in dealing with various types of bisimulations. We intend to study the material of Section 8 in a coalgebraic setting, by looking into the systems based on causal functions and beyond [19,21,11].

Acknowledgements

We thank the anonymous referees of an earlier version of this work for their comments. We are grateful to Ralf Hinze for pointing out to us the work in [8,14,15] and suggesting possible application of our framework to Moessner's theorem which resulted in Section 4. We are grateful for the constructive comments of the three reviewers of the present paper, which certainly have improved the quality of the paper. Niqui was supported by a VENI grant from the Netherlands Organisation for Scientific Research (NWO).

Appendix. Proofs for Moessner's theorem for $k < 4$

We present the proofs that R_0 and R_1 are stream bisimulations. Although each of these facts is proved separately, their proofs follow the same pattern. The proof for R_2 is similar and omitted.

(One can prove Moessner's theorem in general, for all $k \geq 0$, by generalising the proofs for R_0 and R_1, leading to the definition of one (big) relation on streams, which can be shown to be a bisimulation. As we mentioned before, the treatment of the general case falls outside the scope of the present paper.)

Theorem 10. *R_0 is a stream bisimulation.*

Proof. Suppose $\langle \sigma, \tau \rangle \in R_0$. We should prove that $\sigma(0)=\tau(0)$ and $\langle \sigma', \tau' \rangle \in R_0$. We prove this by induction on the structure of R_0. If $\langle \sigma, \tau \rangle \in R_0$ as a result of Clause I0 then trivially $\sigma(0)=\tau(0)$ and $\langle \sigma', \tau' \rangle \in R_0$.

If $\langle \sigma, \tau \rangle \in R_0$ as a result of Clause II0 then the conclusion follows by using the induction hypothesis: in this case $\langle \sigma_0 + \sigma_1, \tau_0 + \tau_1 \rangle \in R_0$ as a result of $\langle \sigma_0, \tau_0 \rangle \in R_0$ and $\langle \sigma_1, \tau_1 \rangle \in R_0$, and by induction hypothesis

$$\sigma_0 = \tau_0, \qquad \sigma_1 = \tau_1,$$
$$\langle \sigma_0', \tau_0' \rangle \in R_0, \qquad \langle \sigma_1', \tau_1' \rangle \in R_0.$$

So by (II0) and the definition of $+$ the conclusion follows.

If $\langle \sigma, \tau \rangle \in R_0$ as a result of Clause III0, i.e., if $\langle \Sigma_2^0(\overline{1}), \Omega \rangle \in R_0$, then

$$(\Sigma_2^0(\overline{1}))(0) = \overline{1}(1) = 1 = \Omega(0);$$

and

$$\begin{aligned}\langle (\Sigma_2^0(\overline{1}))', \Omega' \rangle &= \langle \Sigma_2^0(\overline{1}'') + \overline{\overline{1}(1)}, \Omega + \overline{1} \rangle \\ &= \langle \Sigma_2^0(\overline{1}) + \overline{1}, \Omega + \overline{1} \rangle \in R_0 \quad \text{[by (II0), (III0)]}.\end{aligned}$$

If $\langle \sigma, \tau \rangle \in R_0$ as a result of Clause IV0, i.e., if $\langle \Sigma_2^1(\overline{1}), \Omega \rangle \in R_0$ then

$$\Sigma_2^1(\overline{1})(0) = \overline{1}(0) = 1 = \Omega(0).$$

Finally, for the tails we have

$$\begin{aligned}\langle \big(\Sigma_2^1(\overline{1})\big)', (\Omega)' \rangle &= \langle \Sigma_2^0(\overline{1})' + \overline{\overline{1}(0)}, \Omega + \overline{1} \rangle \\ &= \langle \Sigma_2^0(\overline{1}) + \overline{1}, \Omega + \overline{1} \rangle \in R_0 \quad \text{[by (II0), (III0)]}. \quad \square\end{aligned}$$

Theorem 12. *R_1 is a stream bisimulation.*

Proof. First note that from Theorem 10 and Clause I1 it follows that

$$\langle \Sigma_2^0(\overline{1}), \Omega \rangle \in R_1. \tag{A.1}$$

Suppose $\langle \sigma, \tau \rangle \in R_1$. We should prove that $\sigma(0)=\tau(0)$ and $\langle \sigma', \tau' \rangle \in R_1$. We prove this by induction on the structure of R_1. If $\langle \sigma, \tau \rangle \in R_1$ as a result of Clause I1 then trivially $\sigma(0)=\tau(0)$ and $\langle \sigma', \tau' \rangle \in R_1$.

If $\langle \sigma, \tau \rangle \in R_1$ as a result of Clause II1 then the conclusion follows by using the induction hypothesis: in this case $\langle \sigma_0 + \sigma_1, \tau_0 + \tau_1 \rangle \in R_1$ as a result of $\langle \sigma_0, \tau_0 \rangle \in R_1$ and $\langle \sigma_1, \tau_1 \rangle \in R_1$; and by induction hypothesis

$$\sigma_0 = \tau_0, \qquad \sigma_1 = \tau_1,$$
$$\langle \sigma_0', \tau_0' \rangle \in R_1, \qquad \langle \sigma_1', \tau_1' \rangle \in R_1.$$

So by (II1) and the definition of $+$ the conclusion follows.

If $\langle \sigma, \tau \rangle \in R_1$ as a result of Clause III1, i.e., if $\langle \Sigma_2^0 \Sigma_3^1(\overline{1}), \Omega^{<2>} + \Omega \rangle \in R_1$ then

$$\begin{aligned} \Sigma_2^0 \Sigma_3^1(\overline{1})(0) &= \Sigma_3^1(\overline{1})(1) \\ &= 2 = 1^2 + 1 \\ &= \Omega^{<2>}(0) + \Omega(0) \\ &= (\Omega^{<2>} + \Omega)(0). \end{aligned}$$

For the tails we have

$$\begin{aligned} \langle \big(\Sigma_2^0 \Sigma_3^1(\overline{1})\big)', (\Omega^{<2>} + \Omega)' \rangle &= \langle \Sigma_2^0(\Sigma_3^1(\overline{1}))'' + \overline{\Sigma_3^1(\overline{1})(1)}, (\Omega')^{<2>} + \Omega' \rangle \\ &= \big\langle \Sigma_2^0\big(\Sigma_3^0(\overline{1}') + \overline{1}\big)' + \overline{(\Sigma_3^1(\overline{1}))'(0)}, \\ &\quad (\Omega + \overline{1})^{<2>} + \Omega + \overline{1} \big\rangle \\ &= \big\langle \Sigma_2^0\big((\Sigma_3^0(\overline{1}))' + \overline{1}'\big) + \overline{\Big(\Sigma_3^0(\overline{1}') + \overline{1}\Big)(0)}, \\ &\quad \Omega^{<2>} + \Omega + \Omega + \overline{1} + \Omega + \overline{1} \big\rangle \\ &= \big\langle \Sigma_2^0\big(\Sigma_3^1(\overline{1}'') + \overline{1} + \overline{1}\big) + \overline{\Sigma_3^0(\overline{1})(0) + 1}, \\ &\quad \Omega^{<2>} + \Omega + \Omega + \Omega + \overline{2} \big\rangle \\ &= \big\langle \Sigma_2^0 \Sigma_3^1(\overline{1}) + \Sigma_2^0(\overline{1}) + \Sigma_2^0(\overline{1}) + \overline{2}, \\ &\quad \Omega^{<2>} + \Omega + \Omega + \Omega + \overline{2} \big\rangle \in R_1 \quad \text{[by (II1), (III1), (A.1)]}. \end{aligned}$$

If $\langle \sigma, \tau \rangle \in R_1$ as a result of Clause IV1, i.e., if $\langle \Sigma_2^1 \Sigma_3^2(\overline{1}), \Omega^{<2>} \rangle \in R_1$ then

$$\begin{aligned} \Sigma_2^1 \Sigma_3^2(\overline{1})(0) &= \Sigma_3^2(\overline{1})(0) = 1 \\ &= 1^2 = \Omega^{<2>}(0). \end{aligned}$$

Finally, for the tails we have

$$\begin{aligned} \langle \big(\Sigma_2^1 \Sigma_3^2(\overline{1})\big)', (\Omega^{<2>})' \rangle &= \langle \Sigma_2^0(\Sigma_3^2(\overline{1}))' + \overline{\Sigma_3^2(\overline{1})(0)}, (\Omega')^{<2>} \rangle \\ &= \langle \Sigma_2^0\big(\Sigma_3^1(\overline{1}') + \overline{\overline{1}(0)}\big) + \overline{\overline{1}(0)}, (\Omega + \overline{1})^{<2>} \rangle, \\ &= \big\langle \Sigma_2^0 \Sigma_3^1(\overline{1}) + \Sigma_2^0(\overline{1}) + \overline{1} \\ &\quad \Omega^{<2>} + \Omega + \Omega + \overline{1} \big\rangle \in R_1 \quad \text{[by (II1), (III1), (A.1)]}. \quad \square \end{aligned}$$

References

[1] J.P. Allouche, J. Shallit, Automatic Sequences: Theory, Applications, Generalizations, Cambridge University Press, Cambridge, 2003.
[2] F. Arbab, Reo: a channel-based coordination model for component composition, Math. Structures Comput. Sci. 14 (2004) 329–366.
[3] J. Berstel, C. Reutenauer, Rational series and their languages, in: EATCS Monographs on Theoretical Computer Science, vol. 12, Springer-Verlag, Berlin, 1988.
[4] M. Broy, G. Ştefănescu, The algebra of stream processing functions, Theoret. Comput. Sci. 258 (2001) 99–129.
[5] G. Christol, Ensembles presque periodiques k-reconnaissables, Theoret. Comput. Sci. 9 (1979) 141–145.
[6] N.P. Fogg, Substitutions in dynamics, arithmetics and combinatorics, in: Valérie Berthé, Sébastien Ferenczi, Christian Mauduit, Anne Siegel (Eds.), in: Lecture Notes in Math., vol. 1794, Springer-Verlag, Berlin, 2002.
[7] R.L. Graham, D.E. Knuth, O. Patashnik, Concrete Mathematics. A foundation for computer science, second edition, Addison-Wesley, Reading, Massachusetts, 1994.
[8] R. Hinze, Scans and convolutions: a calculational proof of Moessner's theorem, in: S.B. Scholz (Ed.), IFL 2008 Lecture Notes in Comput. Sci., vol. 5836, Springer-Verlag (to appear), 24 pages.
[9] T.W. Hungerford, Algebra, in: Graduate Texts in Mathematics, vol. 73, Springer-Verlag, New York, 1980, Reprint of the 1974 original.
[10] G. Kahn, The semantics of a simple language for parallel programming, in: Information Processing 74: Proceedings of IFIP Congress 74, in: Stockholm, vol. 74, North Holland Publishing Co., Amsterdam, 1974, pp. 471–475.

[11] J. Kim, Coinductive properties of causal maps, in: J. Meseguer, G. Rosu (Eds.), Proc. of AMAST 2008, in: Lecture Notes in Comput. Sci., vol. 5140, Springer-Verlag, 2008, pp. 253–267.
[12] D. Lucanu, G. Roşu, CIRC: A circular coinductive prover, in: T. Mossakowski, U. Montanari, M. Haveraaen (Eds.), Proc. CALCO 2007, in: Lecture Notes in Comput. Sci., vol. 4624, Springer-Verlag, 2007, pp. 372–378.
[13] R.H. Mak, Design and Performance Analysis of Data-independent Stream Processing Systems, Ph.D. Thesis, Technische Universiteit Eindhoven, 2008.
[14] A. Moessner, Eine Bemerkung über die Potenzen der natürlichen Zahlen, Sitzungsber. Math.-Naturw. Kl. Bayer. Akad. Wiss. München 1951 (1952) 29.
[15] O. Perron, Beweis des Moessnerschen Satzes, Sitzungsber. Math.-Naturw. Kl. Bayer. Akad. Wiss. München 1951 (1952) 31–34.
[16] J.J.M.M. Rutten, Behavioural differential equations: a coinductive calculus of streams, automata, and power series, Theoret. Comput. Sci. 308 (2003) 1–53.
[17] J.J.M.M. Rutten, A coinductive calculus of streams, Math. Structures Comput. Sci. 15 (2005) 93–147.
[18] J.J.M.M. Rutten, A tutorial on coinductive stream calculus and signal flow graphs, Theoret. Comput. Sci. 343 (2005) 443–481.
[19] J.J.M.M. Rutten, Algebraic specification and coalgebraic synthesis of mealy automata, in: L.B. Zhiming Liu (Ed.), Proc. of FACS 2005, in: Electron. Notes Theor. Comput. Sci., vol. 160, Elsevier, 2006, pp. 305–319.
[20] J.J.M.M. Rutten, Rational streams coalgebraically, Log. Methods Comput. Sci. 4 (2008) 1–22.
[21] T. Uustalu, V. Vene, Comonadic notions of computation, in: J. Adámek, C. Kupke (Eds.), Proc. of CMCS 2008, in: Electron. Notes Theor. Comput. Sci., vol. 203(5), Elsevier, 2008, pp. 263–284.
[22] H.S. Wilf, Generating Functionology, second edition, Academic Press Inc., Boston, MA, 1994.